Why Word is a terrible program

Ben Klemens

7 May 2009

[Part one of six]

Yup, this is part one of six: it'll take me six entries to cover the key problems underlying Word's design.

Of course, my interest isn't just in cathartic kvetching, but using Word as a platform for discussing a number of design issues in how we turn data into a final product and efficiently interact with these machines that we sit in front of all day. It's not about what Word got wrong, but how we can do better, both as users and as authors of software.

As for where I got the title of this piece, I stole it from a piece in the New Yorker:

> First of all, it is time to speak some truth to power in this country: *Microsoft Word is a terrible program.*
>
> [...For example,] there is the moment when you realize that your notes are starting to appear in 12-pt. Courier New. Word, it seems, has, at some arbitrary point in the proceedings, decided that although you have been typing happily away in Times New Roman, you really want to be in the default font of the original document. You are confident that you can lick this thing: you painstakingly position your cursor in the Endnotes window (not the text!, where irreparable damage may occur) and click Edit, then the powerful Select All; you drag the arrow to Normal (praying that your finger doesn't lose contact with the mouse, in which case the window will disappear, and trying not to wonder what the difference between Normal and Clear Formatting might be) and then, in the little window to the right, to Times New Roman. You triumphantly click, and find that you are indeed back in Times New Roman but that all your italics have been removed. What about any of this can be considered high-speed?
>
> From *The end matter* by Louis Menand, The New Yorker, issue of 2003-10-06.

Semantic markup The key failing of Word is the difficulty of semantically-oriented editing.

The way most of us format a document in a word processor is to change the formatting of individual elements as we need them. Titles need to be marked in boldface; there needs to be this much space put between paragraphs; the margins should be just so on the cover page. I will call this *literal markup*, where you make changes on the screen until the text looks the way you want it to look.

The alternative is to specify what each element actually *means*, and then worry about how the formatting happens later. Mark the titles as <title>, mark the paragraphs as <text>, and mark the cover page as <cover>. Then, write a style sheet that lists rules that titles should be bold, that text has this much space between paragraphs, and that the cover page's margins are extra-wide. Then, the computer knows to apply the formatting described in the style sheet to your document.

The benefits to semantic markup are immense. First, your boss's boss is going to tell you to change your titles to italics instead of bold as soon as she sees the document. In the semantic system, you change the definition of a title element in one place and you're done; in the literal markup system, you need to go through the entire document and change every title individually—and then repeat when your boss's boss decides that no, you were right, it does look better in bold.

And did you catch the title on page sixty-eight down at the bottom? With semantic markup, because you didn't change fifty points in the document, you don't have to worry about whether you are still consistent or not. More generally, it is by construction impossible to have inconsistent style with a semantic markup scheme, because you define each style exactly once. With literal markup, you need to be on guard for consistency all the time.

In the literal markup world, you wear two hats at the same time: author and typesetter. In the semantic world, you wear the hats one at a time. When working on content, you are not distracted by stylistic junk. Some would describe working only on content and putting off stylistic issues until the very end to be "no fun", but it is certainly more efficient. Of course, you are welcome to play around with the style sheet between writing every other sentence if you so desire.

The document you are working on now is probably not the only document you are writing during your career. Once you have a visual style that you like, you can save those definitions of titles, text, and cover sheet to use on every future document. In the literal markup world, you have to redo the margins on every cover page every time, duplicating a few minutes' effort with every document.[1]

Along a similar vein, your company has a standard letterhead, and may have a graphics department that would prefer all documents to have a consistent style. In semantic-land, your company can distribute a single style sheet and ask that everyone apply it (even if they think it looks ugly; there are always dissenters in this system). In literal markup-land, the graphics department sends out a list of fifty rules everyone must follow when setting up their cover page, wasting everyone's time and guaranteeing that half of the rules won't get followed.

Finally, it is increasingly important that your document be available as a PDF, a web page, and in your company's legacy TPS format. The semantic system, done right, is output-independent. You send the same document to one program that marks up titles appropriate for the printed page, and another that marks it up for the web. If you had to do literal markup appropriate for both the web and paper, it will look terrible in one or the other, and you'd just wind up writing and maintaining two documents.

[1] There are of course tricks, like cutting and pasting the cover sheet from past documents and hoping the formatting follows. Even when they work, you can see that they are still inefficient relative to applying a style sheet.

Semantic markup is hands-down the way to format a document. It doesn't take any more cognitive effort or ability to mark up your title with \title{Intro} or <title>Intro</title> than it does to mark it up to read as **Intro**, so semantic markup provides all of the above benefits over literal markup for basically no cost.

Semantic markup v WYSIWYG Too bad Word is written from the ground up as a program for literal markup. If you read the manual, you will see that there exists a style editor, which claims to allow semantic markup. It lets you define paragraph types and character types, like a title style or a text style.

So the first tip, should you be using Word, is to use the style editor. Avoid hard-coding any sort of formatting; instead, define a style and apply that style.

But it's not entirely that easy, because Word will try its hardest to frustrate you. Word thinks it is smarter than you, so it will often guess the style you mean to be applying to a line, and sometimes revert styles back to where they were. Each element can have only one paragraph and one character style, but there are often reasons to apply multiple styles at once (blockquote on the cover page, italics in a title). Be careful to note where your styles are being saved. If they are being saved to your `normal.dot` template, then when you send your document to your colleagues, your styles won't go with.[2] Best of luck cutting and pasting between documents with different style sheets. There *is* a style organizer that allows you to move style sheet elements from one document to another; it is well-hidden (ask the paperclip for it) but it works. If you want one style for the web version of your document and one for print, you want too much.

The markup is always invisible: you need to click on the item while the style editor is up and then scroll through to see what is highlighted. This may seem trivial, but is frustrating if you have multiple, subtly different styles. And you will, because Word eagerly tries to manage the style list for you. If you italicize an item currently in the title style, then it will autogenerate a title-1 style. Now, when you change all of your titles to non-bold, the lone item in title-1 may or may not follow along.

In short, you *can* use Word for semantic markup, but only with discipline and patience. Word is a literal markup system, and the style editor is your window on Word's internal means of organizing literal markup. It is not a full-blown semantic style sheet, as shown by its little failings above.

The bibliography If you are writing bibliographies by hand, you are wasting your life. Remembering where to put the commas, what to italicize, when to use the full first name and when to use initials, is the sort of work that a computer does easily and that we humans have trouble doing perfectly. Word's demand that users need to hand-edit their bibliographies has no doubt cost the world literally millions of person-hours.

The correct way to do a bibliography is via a database. You provide one entry for each reference, typing out the author, the title, the publisher, et cetera. Then, the computer reads the database and puts the result in your document according to a style sheet such as that published by the University of Chicago or the APA. That is, the best means is via semantic mark-up: you tell the system that "Joe Guzman" is the author,

[2] Tip #2: write yourself a template. Depending on whether the Earth is in a Fire sign or a Water sign, you may need to send the template with your document.

and leave it to the computer to decide whether to print “Guzman, J”, “Guzman, Joe”, “J. Guzman” or what-have-you on the page.

OpenOffice.org wins points for including a bibliography editor. LaTeX includes bibtex. Word does not include one, although you can purchase one from a third party for a hundred dollars or so.[3]

[3]Of course, if you use an add-on like this, you won’t be able to email your document to a colleague (or yourself at another computer) for editing unless the recipient has also paid out the hundred dollars for the same program.